UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/846,067	04/30/2001	Rahul Sharma	SUNMP007	4516

25920 7590 02/24/2006

MARTINE PENILLA & GENCARELLA, LLP
710 LAKEWAY DRIVE
SUITE 200
SUNNYVALE, CA 94085

EXAMINER	
DAO, THUY CHAN	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 02/24/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/846,067	SHARMA ET AL.	
	Examiner	Art Unit	
	Thuy Dao	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *03* MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *25 November 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☐ Claim(s) _____ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3,4,6-8 and 10-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 April 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the amendment filed on November 25, 2005.

2. Claims 1, 3-4, 6-8, 10-20 have been examined. Claims 1, 8, and 15 are independent claims.

Response to Amendments

3. Per Applicant's request, independent claims 1, 8, and 15 have been amended.

4. The objection to drawings Fig. 2-6 and 8-9 is withdrawn in view of Applicant's amendments. The Examiner acknowledges receipt of replacement drawings Fig. 2-6 and 8-9.

5. The objection to the claim 15 is withdrawn in view of Applicant's amendments.

6. Per Provisional Double-Patenting Rejection to claims 1, 3-4, 6-8 and 10-20, Applicants stated that a terminal disclaimer would be filed upon allowance of the present application (Remarks, page 9: 4-12). Thus, the outstanding Double Patenting rejection remains upon such a terminal disclaimer being filed, and it will be reproduced herein of record.

Double Patenting

7. From the record, claims 1, 3-4, 6-8, and 10-18 are under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-13 of a co-pending application 09/833,845. Also, claims 19-20 are under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-13 of the co-pending application in view of Anderson, XP-002239737.

Response to Arguments

8. The Applicant is thanked for thought-out reply. Applicant's arguments concentrate on independent claims 1, 8, and 15.

9. Applicant's arguments filed on November 25, 2005 have been fully considered. However, they are not persuasive.

a) *Hamilton makes no implicit or explicit disclosure of a JAVA module including an entity bean. Rather, Hamilton discloses a database access bridge system and process that enables a two-tire application to operate in a three-tier computer environment without specific programming for the three-tier environment* (Remarks, page 10, lines 9-12).

The Examiner respectfully disagrees with these assertions. Hamilton explicitly disclose an Enterprise Java Bean (EJB) object, which is *a Java module on a server, wherein the Java module is in a middle-tire between a client browser and databases* (e.g., FIG. 2, EJB objects 130 located in Application Server 18, between Client Computer 14 and Database Server 20, col.5: 53-64, col.6: 9-42), and furthermore such EJB objects including entity beans (e.g., col.4: 17-24).

b) *The teachings of Nally do not disclose (and are not related to) how to perform real time modifications to the managed state of a Java-based application with run time object modification and generation, as recited in the claims of the present application, but rather disclose what to do after an application or application user has already made modifications to an Enterprise Java Bean (EJB) and requests to commit the modifications* (Remarks, page 10, lines 17-21).

The Examiner respectfully disagrees with these assertions. Nally discloses real time modifications to the managed state of a Java-based application with run time object modification and generation (e.g., col.1: 53-58, "An EJB's business methods are invoked by sending a message to the EJB's wrapper, where a "wrapper" is the Java term for functionality required to adapt an EJB to its container, and a "container" is the Java terminology for the run-time environment in which an EJB (including the entity bean) is executed"; and

col.4: 25-40, "... such that a particular EJB can be accessed from multiple concurrent and/or nested transactions while ensuring consistency among the transactions and among the nested subtransactions, and while permitting these concurrent and/or nested transactions to have completely independent views of the EJB ... Each transaction within an application has an independent version of the EJB, and

thus an independent transaction tree where the changes made by that transaction are stored during the course of the transaction. Each subtransaction within a nested transaction may also have an independent view of the EJB, by storing that subtransactions' changes at an intermediate level within the tree" (emphasis added)).

c) Applicants added new limitations into independent claims 1, 8, and 15 and argued that Ma, Hamilton, and Nally do not teach or disclose now claimed limitations.

The Examiner respectfully disagrees with these assertions. Ma discloses *a method for upgrading managed state for a JAVA based application* (e.g., FIG. 3 and related text, column 6, lines 31-67), the method comprising:

generating an upgraded state object (e.g., column 8, lines 55; FIG. 3 and items 68', 68, and related text), wherein the upgraded state object is generated by upgrading a physical schema using data stored in a repository that is part of the databases (e.g., FIG. 3, item 62 and related text, column 4, liens 42-48);

transferring the state stored in the original state object to the upgraded stated object (e.g., column 9, lines 20-27; column 11, lines 25-40);

providing state management for an original entity component *using the upgraded state object* (e.g., FIG. 8, items 152, 144, and related text).

Ma does not explicitly disclose *executing a JAVA module on a server, wherein the JAVA module is in a middle-tier between a client browser and databases, the JAVA module includes at least one original entity bean and at least one original state object in communication with the original entity bean, the original state object storing a state of the original entity bean.*

However, in an analogous art of executing a JAVA module (EJB) in multi-tier computer environment, Hamilton discloses executing an EJB (JAVA module) on a server, wherein the EJB is in a middle-tier between a client browser and databases (e.g., FIG. 2, EJB objects 130 located in Application Server 18, between Client Computer 14 and Database Server 20, col.5: 53-64, col.6: 9-42), and furthermore such EJB objects including entity beans (e.g., col.4: 17-24); and

column 1, lines 47-52, "In a three-tier environment, a client system (first tier) has a GUI that communicates with an application running on an application server (second tier) which in turn communicates with a database server (third tier) for access to and storage of data").

Ma's discloses a run-time object-updating system updating a distributed-object client-server application with client objects and server objects (e.g., DCOM and CORBA as distributed computing middle-ware) in a Windows or Solaris computer environment (FIGs. 3, 6, and related text, e.g., col.6: 31-67). Hamilton's discloses a system and method enabling a two-tier computer application to operate in a three-tier environment (e.g., col.2: 30-37), and particularly using EJB components/objects as distributed computing middle-ware (e.g., FIG. 2, EJB objects 130 located in Application Server 18, between Client Computer 14 and Database Server 20, col.5: 53-64, col.6: 9-42).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to modify the teachings of Ma by having said three-tier environment as taught by Hamilton. One would have been motivated to do so because the three-tier environments enable business applications on the second tier to be modified without having to substantially modify each client system as well as provide more flexibility and efficiency as suggested by Hamilton (e.g., column 1, lines 52-60).

Although Ma and Hamilton used EJB including entity bean but they do not explicitly disclose that includes at least one original state object in communication with the original entity bean, the original state object storing a state of the original entity bean.

However, in an analogous art of managing Enterprise JavaBeans (EJBs) and maintaining an integrity of the EJBs, Nally discloses said an Enterprise JavaBean (EJB) including an original entity bean and at least one original state object in communication with the original entity bean, the original state object storing a state of the original entity bean (e.g., column 1, lines 52-53, "For an EJB, the executable business logic is stored within the entity bean";

FIG. 5, column 13, lines 29-34, "The present invention adds another logical substructure to each version 520, beyond what is specified in the EJB

specification, which is version status data 530. This version status data 530 is used internally by an implementation of the present invention to keep track of information about this version's status, such as whether the version has been modified" (emphasis added)).

Nally further discloses the EJB (*JAVA module*) *including the upgraded state object and the original state object is upgraded in the JAVA module* (e.g., FIG. 5, column 13, lines 35-43, "... each time a method in the logic 540 of the bean changes the bean version (i.e. the instance data 550 for the version), this version status data 530 stores the knowledge that the bean has been modified in order to later communicate this information upward in the transaction tree 400 upon a commit (so that the root transaction will detect that the changes must be reflected in persistent storage"; column 13, lines 45-65; and column 14, lines 27-61).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine the teachings of Nally into that of Ma and Hamilton. One would have been motivated to do so to ensure the consistency of Enterprise JavaBeans among multiple applications that are concurrently accessing the same EJB, enable nested transactions may be used in an application, and detect conflicts when a user or application attempts to commit changes to EJBs in a persistent data store as suggested by Nally (e.g., column 4, lines 45-67; column 9, line 48 to column 10, line 21).

d) Applicants asserted that neither Ma, Hamilton, nor Nally discloses or teach the claimed limitations in independent claims 1, 8, and 15 (Remarks, pages 9-11).

In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See In re Keller, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); In re Merck & Co., 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

e) Applicants asserted that no suggestion or motivation could be found to apply the principles taught in Hamilton and Nally to the claims of the present application and a prima facie cased of obviousness has not been established (Remarks, page 12, lines 7-10, lines 22-24).

In response to applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See In re McLaughlin, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

Claim Rejections – 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 1, 3-4, 6-8, and 10-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ma (art of record, US Patent No. 5,920,725) in view or Hamilton (art of record, US Patent No. 6,889,227) and further in view of Nally (art of record, US Patent No. 6,298,478).

Claim 1:

Ma discloses *a method for upgrading managed state for a JAVA based application* (e.g., FIG. 3 and related text, column 6, lines 31-67), the method comprising:

generating an upgraded state object (e.g., column 8, lines 55; FIG. 3 and items 68', 68, and related text), wherein the upgraded state object is generated by

upgrading a physical schema using data stored in a repository that is part of the databases (e.g., FIG. 3, item 62 and related text, column 4, liens 42-48);

transferring the state stored in the original state object to the upgraded stated object (e.g., column 9, lines 20-27; column 11, lines 25-40);

providing state management for an original entity component *using the upgraded state object* (e.g., FIG. 8, items 152, 144, and related text).

Ma does not explicitly disclose *executing a JAVA module on a server, wherein the JAVA module is in a middle-tier between a client browser and databases, the JAVA module includes at least one original entity bean and at least one original state object in communication with the original entity bean, the original state object storing a state of the original entity bean.*

However, in an analogous art of executing a JAVA module (EJB) in multi-tier computer environment, Hamilton discloses executing an EJB (JAVA module) on a server, wherein the EJB is in a middle-tier between a client browser and databases (e.g., FIG. 2, EJB objects 130 located in Application Server 18, between Client Computer 14 and Database Server 20, col.5: 53-64, col.6: 9-42), and furthermore such EJB objects including entity beans (e.g., col.4: 17-24); and

column 1, lines 47-52, "In a three tier environment, a client system (first tier) has a GUI that communicates with an application running on an application server (second tier) which in turn communicates with a database server (third tier) for access to and storage of data").

Ma's discloses a run-time object-updating system updating a distributed-object client-server application with client objects and server objects (e.g., DCOM and CORBA as distributed computing middle-ware) in a Windows or Solaris computer environment (FIGs. 3, 6, and related text, e.g., col.6: 31-67). Hamilton's discloses a system and method enabling a two-tier computer application to operate in a three-tier environment (e.g., col.2: 30-37), and particularly using EJB components/objects as distributed computing middle-ware (e.g., FIG. 2, EJB objects 130 located in Application Server 18, between Client Computer 14 and Database Server 20, col.5: 53-64, col.6: 9-42).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to modify the teachings of Ma by having said three-tier environment as taught by Hamilton. One would have been motivated to do so because the three-tier environments enable business applications on the second tier to be modified without having to substantially modify each client system as well as provide more flexibility and efficiency as suggested by Hamilton (e.g., column 1, lines 52-60).

Although Ma and Hamilton used EJB including entity bean but they do not explicitly disclose that includes at least one original state object in communication with the original entity bean, the original state object storing a state of the original entity bean.

However, in an analogous art of managing Enterprise JavaBeans (EJBs) and maintaining an integrity of the EJBs, Nally discloses said an Enterprise JavaBean (EJB) including an original entity bean and at least one original state object in communication with the original entity bean, the original state object storing a state of the original entity bean (e.g., column 1, lines 52-53, "For an EJB, the executable business logic is stored within the entity bean";

FIG. 5, column 13, lines 29-34, "The present invention adds another logical substructure to each version 520, beyond what is specified in the EJB specification, which is version status data 530. This version status data 530 is used internally by an implementation of the present invention to keep track of information about this version's status, such as whether the version has been modified" (emphasis added)).

Nally further discloses the EJB *(JAVA module) including the upgraded state object and the original state object is upgraded in the* EJB *(JAVA module)* (e.g., FIG. 5, column 13, lines 35-43, "... each time a method in the logic 540 of the bean changes the bean version (i.e. the instance data 550 for the version), this version status data 530 stores the knowledge that the bean has been modified in order to later communicate this information upward in the transaction tree 400 upon a commit (so that the root transaction will detect that the changes must be reflected in persistent storage"; column 13, lines 45-65; and column 14, lines 27-61).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine the teachings of Nally into that of Ma and Hamilton. One would have been motivated to do so to ensure consistency of Enterprise JavaBeans among multiple applications that are concurrently accessing the same EJB, enable nested transactions may be used in an application, and detect conflicts when a user or application attempts to commit changes to EJBs in a persistent data store as suggested by Nally (e.g., column 4, lines 45-67; column 9, line 48 to column 10, line 21).

Claim 3:

The rejection of base claim 1 is incorporated. Ma also discloses *comprising the operation of managing the state of the upgraded entity bean using the upgraded state object* as addressed in claim 1 above (e.g., FIG. 8, items 146, 147, 152 and related text, column 11, lines 1-55, "Object adaptor 80 also updates the client class definitions. An invalidate notice is sent to the client's object cache, invalidating the client. However, old client object 144 has already begun to reference new server object 152, which was created based on new classes 140 when old client object 144 instantiated the server object. Since old client object 144 has an old interface while new server object 152 has a new interface, an error occurs. An error is signaled from new server object 152 to old client object 144. An error-handling routine in old client object 144 determines that old client object 144 is invalid. The error-handling routine then re-loads old client object 144, which becomes new client object 147. The state of old client object 144 is transferred to new client object 147 when it is created from new classes 140. New client object 147 can then re-access or re-create new server object 152 and use the new interface").

Claim 4:

The rejection of base claim 3 is incorporated. Ma also discloses *both the original entity bean and the original state object are disabled* (e.g., column 4, lines 59-63, "Other server objects and other client objects continue to run while the object adaptor invalidates the obsolete objects and creates the new server objects and the new client

objects. Thus the distributed-object client-server application is updated while running"; and

column 5, lines 17-21, "A cache invalidation means in the object adaptor invalidates copies of the obsolete objects by invalidating obsolete class definitions indexed by the client caches. Thus obsolete client objects and classes are invalidated through the client caches").

Claim 6:

The rejection of base claim 4 is incorporated. Ma also discloses *functionality of the JAVA module is not disrupted when the upgraded state object is generated* (e.g., page 1, paragraph [57], "A distributed client-server application is modified while running. The application is not stopped so that updating of objects is transparent. A meta server catalogs all object classes for both the server and the clients. Modifications are specified by a run-time update tool and converted to change commands"; and

column 4, lines 59-63, "Other server objects and other client objects continue to run while the object adaptor invalidates the obsolete objects and creates the new server objects and the new client objects. Thus the distributed-object client-server application is updated while running").

Claim 7:

The rejection of base claim 4 is incorporated. Ma also discloses *functionality of the JAVA application is not disrupted when the JAVA module is upgraded* (e.g., page 1, paragraph [57], "A distributed client-server application is modified while running. The application is not stopped so that updating of objects is transparent. A meta server catalogs all object classes for both the server and the clients. Modifications are specified by a run-time update tool and converted to change commands"; and

column 4, lines 59-63, "Other server objects and other client objects continue to run while the object adaptor invalidates the obsolete objects and creates the new server objects and the new client objects. Thus the distributed-object client-server application is updated while running").

Claim 8:

Ma discloses *a JAVA platform capable of performing an online upgrade on a JAVA application, the JAVA platform comprising:*

a repository that is part of the databases and having upgraded class files for the original entity component *and upgraded class files for the original state object* (e.g., FIG. 3 item 62 Repository and related text, column 6, lines 39-51);

wherein the original state object is upgraded by generating an upgraded state object using upgraded class files from the repository, and transferring the state stored in the original state object to the upgraded state object (e.g., column 11, lines 25-40); and

an upgrade entity component *is created using data from the repository as the JAVA platform is upgraded* (e.g., column 6, lines 19-23; column 7, lines 19-39; and column 9, lines 20-22).

Ma does not explicitly disclose *executing a JAVA module on a server, wherein the JAVA module is in a middle-tier between a client browser and databases, the JAVA module includes at least one original entity bean and at least one original state object in communication with the original entity bean, the original state object storing a state of the original entity bean.*

However, in an analogous art of executing a JAVA module (EJB) in multi-tier computer environment, Hamilton discloses executing an EJB (JAVA module) on a server, wherein the EJB is in a middle-tier between a client browser and databases (e.g., FIG. 2, EJB objects 130 located in Application Server 18, between Client Computer 14 and Database Server 20, col.5: 53-64, col.6: 9-42), and furthermore such EJB objects including entity beans (e.g., col.4: 17-24); and

column 1, lines 47-52, "In a three tier environment, a client system (first tier) has a GUI that communicates with an application running on an application server (second tier) which in turn communicates with a database server (third tier) for access to and storage of data").

Ma's discloses a run-time object-updating system updating a distributed-object client-server application with client objects and server objects (e.g., DCOM and CORBA as distributed computing middle-ware) in a Windows or Solaris computer environment (FIGs. 3, 6, and related text, e.g., col.6: 31-67). Hamilton's discloses a system and method enabling a two-tier computer application to operate in a three-tier environment (e.g., col.2: 30-37), and particularly using EJB components/objects as distributed computing middle-ware (e.g., FIG. 2, EJB objects 130 located in Application Server 18, between Client Computer 14 and Database Server 20, col.5: 53-64, col.6: 9-42).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to modify the teachings of Ma by having said three-tier environment as taught by Hamilton. One would have been motivated to do so because the three-tier environments enable business applications on the second tier to be modified without having to substantially modify each client system as well as provide more flexibility and efficiency as suggested by Hamilton (e.g., column 1, lines 52-60).

Although Ma and Hamilton used EJB including entity bean but they do not explicitly disclose that includes at least one original state object in communication with the original entity bean, the original state object storing a state of the original entity bean.

However, in an analogous art of managing Enterprise JavaBeans (EJBs) and maintaining an integrity of the EJBs, Nally discloses said an Enterprise JavaBean (EJB) including an original entity bean and at least one original state object in communication with the original entity bean, the original state object storing a state of the original entity bean (e.g., column 1, lines 52-53, "For an EJB, the executable business logic is stored within the entity bean";

FIG. 5, column 13, lines 29-34, "The present invention adds another logical substructure to each version 520, beyond what is specified in the EJB specification, which is version status data 530. This version status data 530 is used internally by an implementation of the present invention to keep track of information about this version's status, such as whether the version has been modified" (emphasis added)).

Nally further discloses the EJB (*JAVA module) including the upgraded state object and the original state object is upgraded in the* EJB (*JAVA module)* (e.g., FIG. 5, column 13, lines 35-43, "According to the novel features of the present invention, each time a method in the logic 540 of the bean changes the bean version (i.e. the instance data 550 for the version), this version status data 530 stores the knowledge that the bean has been modified in order to later communicate this information upward in the transaction tree 400 upon a commit (so that the root transaction will detect that the changes must be reflected in persistent storage"; column 13, lines 45-65; and column 14, lines 27-61).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine the teachings of Nally into that of Ma and Hamilton. One would have been motivated to do so to ensure consistency of Enterprise JavaBeans among multiple applications that are concurrently accessing the same EJB, enable nested transactions may be used in an application, and detect conflicts when a user or application attempts to commit changes to EJBs in a persistent data store as suggested by Nally (e.g., column 4, lines 45-67; column 9, line 48 to column 10, line 21).

Claim 10:

The rejection of base claim 8 is incorporated. Ma also discloses *the state of the upgraded entity bean is managed using the upgraded state object.*

Claim 10 recites the same limitations as those of claim 3, wherein all claimed limitations have been addressed and/or set forth above. Therefore, as the references teach all of the limitations of claim 3, they also teach all of the limitations of claim 10.

Claim 11:

The rejection of bas claim 10 is incorporated. Ma also discloses *both the original entity bean and the original state object are disabled.*

Claim 11 recites the same limitations as those of claim 4, wherein all claimed limitations have been addressed and/or set forth above. Therefore, as the references teach all of the limitations of claim 4, they also teach all of the limitations of claim 11.

Claim 12:

The rejection of base claim 8 is incorporated. Ma also discloses *the upgraded state object is generated by upgrading a physical schema using data stored in the repository* (e.g., column 5, line 66 to column 6, line 6, "The inventors have realized that central repository techniques can be used for cataloging and storage of program elements. The inventors construct a meta server that stores the object descriptors for all programming objects and uses a database to provide non-volatile storage to this repository of object descriptors. New instances of objects are created from the object description fetched from the meta server's database"; and

column 6, lines 3-51, "Meta server's database 62 is a non-volatile repository of all object class definitions for the distributed application. Even remote objects that are instantiated only on remote clients and not on the server have their class definitions stored persistently in meta server's database 62. The class definitions include the interface definitions, attributes, procedures, and grouping of objects into folders. The class definitions contain the blueprint of objects that are inherited by each instance of an object generated or instantiated. Every time an object is instantiated, the object class definition in meta server 70 serves as the blueprint, although a cache of that object class definition may be used to speed up object creation").

Claim 13:

The rejection of base claim 12 is incorporated. Ma also discloses *functionality of the JAVA module is not disrupted when the JAVA module is upgraded.*

Claim 13 recites the same limitations as those of claim 6, wherein all claimed limitations have been addressed and/or set forth above. Therefore, as the references teach all of the limitations of claim 6, they also teach all of the limitations of claim 13.

Claim 14:

The rejection of base claim 13 is incorporated. Ma also discloses *functionality of the JAVA application is not disrupted when the JAVA module is upgraded.*

Claim 14 recites the same limitations as those of claim 7, wherein all claimed limitations have been addressed and/or set forth above. Therefore, as the references teach all of the limitations of claim 7, they also teach all of the limitations of claim 14.

Claims 15-18:

Claims 15-18 recite the same limitations as those of the method claims 1, 3-4, 6-7, and 12, wherein all claimed limitations have been addressed and/or set forth above. Therefore, as the references teach all of the limitations of claims 1, 3-4, 6-7, and 12, they also teach all of the limitations of claims 15-18.

12. Claims 19 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ma-Hamilton-Nally as applied to claim 18 above, and further in view of Anderson (art of record, NPL reference).

Claim 19:

The rejection of base claim 18 is incorporated. Ma, Hamilton, and Nally do not explicitly disclose *the original state object and the upgraded state object are respectively classified into a particular state management unit.*

However, in an analogous art, Anderson discloses the original state object and the upgraded state object are respectively classified (e.g., page 3, left column, section TOOLS, "JDrums provides a tool for automatically generating a conversion skeleton class. which includes all imports, fields and methods that are used to refer to the old and new version of a specific class. A class might have a more or less complex relationship to other classes. The generating tool should in any case figure this out and reflect it in the conversion class.

The tool is connected to the JDrums-enabled JVM through the communication layer and can thereby retrieve information about the old class that is to

be updated. Also, it gets information about the enhanced, new class, from a local source.

By automatically generating a conversion class, we minimize the risk of using non-matching fields and methods in our mirror of the old and new class. For instance if the wrong field is used. the compiler will complain")

into a particular state management unit (e.g., pages 2-3, section STRATEGY, "To be able to perform a conversion it is necessary that we have all the information about the component that we intend to update. This is achieved by consulting the running JVM. Alternatively we could have inspected a non-executing entity, e.g. the actual source code. The drawback doing this is that in some cases you might not have access to this entity. We also need information about the updated component. The information gathered so far can be used to generate a skeleton describing attributes of both components. This constitutes, together with the updated component, the conversion package. In the prototype the updates are restricted to the methods and attributes of the class in the conversion packaqe, i.e. members of inherited classes are not accessible, neither is the inheritance graph. The restriction lies not in the JDrums JVM, but in tool which generates the skeleton.

When the conversion package has been created it is sent to the JDrums-enabled JVM through the communication layer. The JVM has been specifically equipped so that it uses the conversion package to reconfigure the running application").

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine the teachings of Ma-Hamilton-Nally into that of Anderson. One would have been motivated to do so to enhance the system as suggested by Anderson (e.g., page 1, column 1, section Introduction).

Claim 20:

The rejection of base claim 19 is incorporated. Claim 20 recites the limitations, wherein all claimed limitations have been addressed and/or set forth in claim 19 above.

Therefore, as the references teach all of the limitations of claim 19, they also teach all of the limitations of claim 20.

Conclusion

13. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure:

"A Detailed Comparison of CORBA, DCOM and JAVA/RMI", Gopalan S. Raj, September 1998, discloses CORBA and DCOM as distributed objects interact with run-time services in server-client computer environments.

"A Detailed Comparison of Enterprise JavaBeans (EJB) & The Microsoft Transaction Server (MTS) Models", Gopalan S. Raj, May 1999, discloses EJB requires a controlled runtime environment. EJB components run inside an EJB Container and these middleware components are kept not idle and working to their maximum potential.

US Patent No. 6,944,680 discloses container-managed persistence and Bean-managed persistence. For Container-managed persistence, the EJB container is responsible for saving the Bean's state. Since it is container-managed, the implementation is independent of the data source. The container-managed fields need to be specified in the Deployment Descriptor and the container automatically handles the persistence. For Bean-managed persistence the Entity Bean is directly responsible for saving its own state.

US Patent No. 6,457,065 discloses logical partitioning of an EJB into an EJB Object and an entity Bean, and the logical structure that may be used to replicate versions of an entity Bean in a transaction view.

US Patent No. 6,684,387 discloses method and apparatus for verifying Enterprise JavaBeans.

14. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

14. Any inquiry concerning this communication should be directed to examiner Thuy Dao (Twee), whose telephone is (571) 272 8570. The examiner can normally be reached on Monday – Friday from 6:30AM to 3:00PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam, can be reached at (571) 272 3695.

The fax phone number for the organization where this application or proceeding is assigned is (703) 872 9306.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is (571) 272 2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

T. Dao

TUAN DAM
SUPERVISORY PATENT EXAMINER